

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-015647

(43)Date of publication of application : 22.01.1999

(51)Int.Cl.

G06F 9/06

G06F 9/06

(21)Application number : 09-169939

(71)Applicant : HITACHI LTD

(22)Date of filing : 26.06.1997

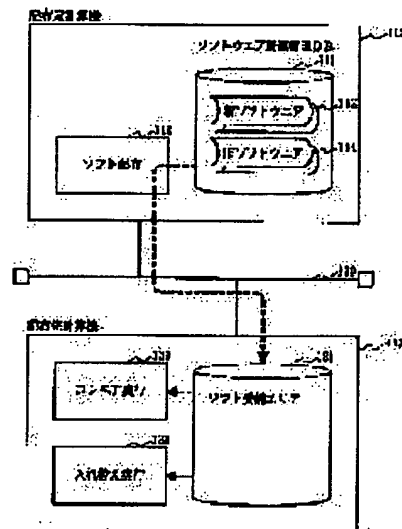
(72)Inventor : NISHIJIMA HIDEJI
FUJIWARA KAZUNORI
KONO KATSUMI
WATAYA HIROSHI

(54) SOFTWARE MAINTENANCE METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a means for ensuring the restoration of the software of a former (present) version for the failure of the exchange of software from the present version to a new version in the maintenance of software.

SOLUTION: A comparison execution means 132 is provided for a distribution destination computer in a state where the new version 113 and the former version 114 of software resource are stored in the prescribed storage area 131 of the distribution destination computer. The comparison execution means verifies identity between an execution module which is artificially generated from the software of the former version and an execution module (that is, the feasible form of the software of former version) which is previously registered before the exchange of software from the former version to the new version. When a verified result is satisfactory, a processing 133 for exchanging software to that of new version is executed. When the verified result is not satisfactory, the exchange processing 133 is stopped.



LEGAL STATUS

[Date of request for examination] 26.05.2004

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision]

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平11-15647

(43)公開日 平成11年(1999) 1月22日

(51)Int.Cl.*

G 0 6 F 9/06

識別記号

4 1 0

5 4 0

F I

G 0 6 F 9/06

4 1 0 P

5 4 0 C

審査請求 未請求 請求項の数12 O L (全 9 頁)

(21)出願番号

特願平9-169939

(22)出願日

平成9年(1997) 6月26日

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目 6 番地

(72)発明者 西島 英児

神奈川県川崎市麻生区王禅寺1099番地 株

式会社日立製作所システム開発研究所内

(72)発明者 藤原 和紀

茨城県日立市大みか町五丁目 2 番 1 号 株

式会社日立製作所大みか工場内

(72)発明者 河野 克己

神奈川県川崎市麻生区王禅寺1099番地 株

式会社日立製作所システム開発研究所内

(74)代理人 弁理士 小川 勝男

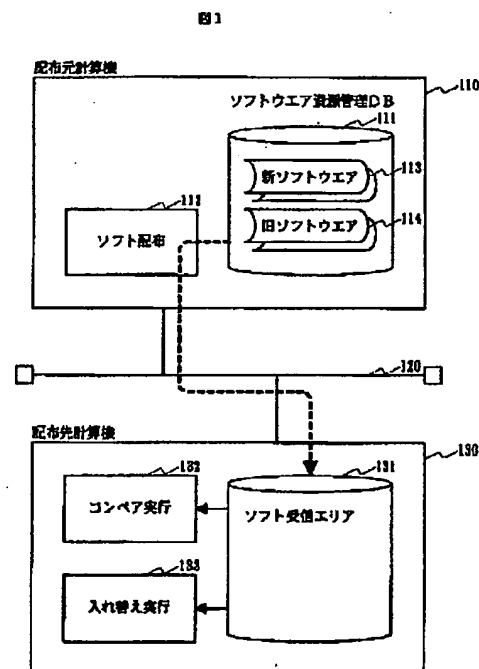
最終頁に続く

(54)【発明の名称】 ソフトウェア保守方法

(57)【要約】

【課題】ソフトウェア保守において、現行バージョンから新バージョンへのソフトウェアの入れ替えの失敗に備えて、旧（現行）バージョンのソフトウェアへの復旧を保証（常に元通りに戻せられるということ）しておく手段を提供することである。

【解決手段】ソフトウェア資源の新バージョン113と旧バージョン114が配布先計算機の所定の記憶領域131に格納された状態において、配布先計算機にコンペア実行手段132を設けておき、該ソフトウェアを旧バージョンから新バージョンに入れ替える前に、コンペア実行手段が旧バージョンのソフトウェアから疑似的に生成される実行モジュールと既に登録されてある実行モジュール（旧バージョンのソフトウェアの実行可能形式を指す）間の同一性を検証する。検証結果が良好の時には、新バージョンのソフトウェアに入れ替える処理133に移行するが、検証結果が不良の時には、このような入れ替え処理133を中止する。



【特許請求の範囲】

【請求項1】配布元計算機が新バージョンのソフトウェア資源を配布先計算機へ格納し、該配布先計算機が新バージョンと旧バージョンのソフトウェア資源を保持しており、該ソフトウェア資源の各バージョンがソースファイルまたはオブジェクトまたはロードモジュールおよびインストールスクリプトまたはコンペアスクリプトからなり、該ソフトウェア資源のうちのインストールスクリプトを起動して実行可能形式を生成するための入れ替え実行手段を有する場合において、該配布先計算機にコンペア実行手段を設けて、該入れ替え実行手段が起動する前に、該コンペア実行手段が該コンペアスクリプトを起動することによって該旧バージョンのソフトウェア資源から生成される実行可能形式と現行稼働状態の実行可能形式間の一致性を検証した後に、検証結果に従って該入れ替え実行手段が実行可能形式の入れ替えを実施することを特徴とするソフトウェア保守方法。

【請求項2】請求項1項記載の配布先計算機において、前記コンペア実行手段が該インストールスクリプトを解析してコンペアコマンドを生成・実行することによって該旧バージョンのソフトウェア資源から生成される実行可能形式と現行稼働状態の実行可能形式間の一致性を検証した後に、検証結果に従って該入れ替え実行手段が実行可能形式の入れ替えを実施することを特徴とするソフトウェア保守方法。

【請求項3】請求項1項記載の配布先計算機において、前記コンペア実行手段がインストール先を一時的なエリアに指定して該インストールスクリプトを起動した後に、該一時的なエリアに生成された実行可能形式と現行稼働状態の実行可能形式間の一致性を検証した後に、検証結果に従って該入れ替え実行手段が実行可能形式の入れ替えを実施することを特徴とするソフトウェア保守方法。

【請求項4】前記コンペア実行手段が配布元計算機から配布先計算機へソフトウェア資源を格納した後に起動されること、を特徴とする請求項1～3項記載のソフトウェア保守方法。

【請求項5】前記コンペア実行手段が配布先計算機の立ち上げ後に起動されること、を特徴とする請求項1～3項記載のソフトウェア保守方法。

【請求項6】前記コンペア実行手段が配布先計算機の立ち下げ時に起動されること、を特徴とする請求項1～3項記載のソフトウェア保守方法。

【請求項7】前記コンペア実行手段が指定された日付や時刻で起動されること、を特徴とする請求項1～3項記載のソフトウェア保守方法。

【請求項8】前記コンペア実行手段が低い優先度で起動されること、を特徴とする請求項1～3項記載のソフトウェア保守方法。

【請求項9】前記コンペア実行手段による検証結果が全

て一致の時に限り前記入れ替え実行手段を実行し、該検証結果が少なくとも1つの不一致の時には前記入れ替え実行手段を実行しないこと、を特徴とする請求項1～3項記載のソフトウェア保守方法。

【請求項10】前記コンペア実行手段による検証結果が一致となったソフトウェア資源に対して前記入れ替え実行手段を実行すること、を特徴とする請求項1～3項記載のソフトウェア保守方法。

【請求項11】前記コンペアスクリプトが前記ソフトウェア資源を元に一時的に生成する実行可能形式と現行稼働状態の実行可能形式間の一致性を比較するコマンド群からなること、を特徴とする請求項1～3項記載のソフトウェア保守方法。

【請求項12】前記インストールスクリプトが前記ソフトウェア資源を元に実行可能な形式を生成するコマンド群からなり、ソフトウェア種別・ソフトウェア名・ソフトウェア資源名・インストールオプション別に分類されたフォーマットからなること、を特徴とする請求項1～3項記載のソフトウェア保守方法。

【発明の詳細な説明】**【0001】**

【発明の属する技術分野】本発明は、分散配置された複数の配布先計算機へのソフトウェア保守に係り、特に、各配布先計算機へ新バージョンのプログラムを入れ替える前に、万が一入れ替えが失敗した時にシステム稼働の異常事態を避けるために、元の状態に確実に復旧できるという保証をとっておくのに好適なソフトウェア保守方法に関する。

【0002】

【従来の技術】従来の技術は、特開平第5-108317号「ネットワークシステム及びそのソフトウェア管理方法」に示されている。

【0003】従来の技術は、配布元計算機が事前に準備された新バージョンのソフトウェアを保持しており配布先計算機に新バージョンの格納エリアにダウンロードした後に、配布先計算機が新バージョンの格納エリア内のソフトウェアのうちのインストールスクリプトを起動することによって現行バージョンから新バージョンへのソフトウェアの入れ替えを実施する。新バージョンへのソフトウェアの入れ替えが成功した時には、新バージョンの格納エリアを現行バージョンの格納エリアに移動させる。一方、新バージョンへのソフトウェアの入れ替えが失敗した時には、配布先計算機が旧（現行）バージョンの格納エリア内のソフトウェアのうちのインストールスクリプトを起動することによって旧（現行）バージョンのソフトウェアへの復旧を実施する。常に、旧（現行）バージョンの格納エリアには、前回の入れ替えの実施で成功した、旧（現行）バージョンのソフトウェアが格納されている。

【0004】

【発明が解決しようとする課題】従来の技術は、万が一現行バージョンから新バージョンへのソフトウェアの入れ替えが失敗した時に、旧（現行）バージョンのソフトウェアへの復旧が保証（常に元通りに戻せられるということ）されていないために、旧（現行）バージョンのソフトウェアへの復旧も失敗することがあり、結局、配布先計算機のソフトウェアシステムの動作不能によって業務の支障を招く恐れがある。具体的な一例を以下に説明する。

【0005】例えば、リアルタイム制御計算機に代表される常駐型タスクは、主メモリ内にローディングされることによってインストールが完了するものであり、次のようなインストールの制約がある。なお、タスクとは同時に起動できる実行モジュール数が1つに限られている実行モジュールの種別を指す。

【0006】（A1）メモリ内の実行モジュールをバックアップする手段がない

（A2）メモリ内の他の実行モジュール間とリンケージをとるためのインストールスクリプトが必要となる。

【0007】このようなインストールの制約によって、次のような問題点がある。

【0008】「タスクを新バージョンに入れ替えた時に、これが万が一失敗した時に、元の旧バージョンへ確実に戻すことができない」この問題の発生する原因には、旧バージョンの入れ替え直後と新バージョンの入れ替え直後間で状況の変化があるため、具体的には次のような要因があげられる。

【0009】（B1）ローディングパラメータの違い

（B2）メモリ使用状況の変化

（B3）パッチによるメモリ空間のアドレスとデータ関係の変化

このような要因がある限り、ソフトウェア保守の高信頼性を図るためには、次の課題を解決する必要がある。

【0010】「メモリ内へ常駐する実行モジュールが、入れ替える前の状態と、復旧した後で、アドレス空間およびデータが一致化されていることを保証しなければならない」

以上のように、本発明の目的は、現行バージョンから新バージョンへのソフトウェアの入れ替えの失敗に備えて、旧（現行）バージョンのソフトウェアへの復旧を保証（常に元通りに戻せられるということ）しておく手段を提供することである。

【0011】

【課題を解決するための手段】ソフトウェアの新バージョンと旧バージョンが事前に配布先計算機の所定の記憶領域に格納された状態において、配布先計算機にコンペア実行手段を設けておき、該ソフトウェアを旧バージョンから新バージョンに入れ替える前に、コンペア実行手段が旧バージョンのソフトウェアから疑似的に生成される実行モジュールと既に登録されてある実行モジュール

（旧バージョンのソフトウェアの実行可能形式を指す）間の同一性を検証する。

【0012】検証結果が良好の時には、新バージョンのソフトウェアに入れ替える処理に移行するが、検証結果が不良の時には、このような入れ替え処理を中止する。

【0013】コンペア実行手段は、旧バージョンのソフトウェアがソースファイルまたはオブジェクトやライブラリおよびインストールスクリプトに加えてコンペアスクリプトを含んでいる時には、コンペアスクリプトを起動する。または、旧バージョンのソフトウェアがソースファイルまたはオブジェクトやライブラリおよびインストールスクリプトのみを含んでいる時には、インストールスクリプトを解析して、ソフトウェア種別に従ってコンペアコマンドを生成・実行する。ここで、コンペアスクリプトとは、旧バージョンのソフトウェアから疑似的に生成される実行モジュールと既に登録されてある実行モジュール間を比較するためのコンペアコマンドを記述したものである。インストールスクリプトとは、旧バージョンのソフトウェアから実行モジュールを生成するためのコンパイル・リンク・ロードコマンドを記述したものである。

【0014】ソフトウェアが複数ある場合には、コンペア実行手段は、複数のソフトウェアを実行対象とする。

【0015】前記コンペア実行手段によって、旧バージョンのソフトウェアから疑似的に生成される実行モジュールと既に登録されてある実行モジュール（旧バージョンのソフトウェアの実行可能形式を指す）間の同一性を検証しておくため、万が一、該ソフトウェアを旧バージョンから新バージョンに入れ替える処理が失敗した時でも、旧バージョンのソフトウェアへの復旧処理は、入れ替える前の実行モジュールに元通りに戻すことが可能となる。

【0016】このような元通りに戻せられるという高信頼なソフトウェア保守の実現によって、常にソフトウェアは正常な動作状態を維持することができ、ソフトウェアシステムの安全な稼働を保証させることができる。

【0017】

【発明の実施の形態】以下、本発明を実現するための実施例を図面により説明する。

【0018】〔1〕実施例1

（A）ハード・ソフトウェア構成例

図1に本発明を実現するためのハード・ソフトウェア構成の一例を示す。図1は、ハードウェア構成として、配布元計算機110と配布先計算機130がネットワーク120に接続された構成を示している。配布先計算機130は1つに限定されるものではなく複数のものがネットワーク120に接続されていてもかまわない。配布元計算機110はソフトウェア資源管理データベース111およびソフト配布手段112を有する。一方、配布先計算機130はソフト受信エリア131、コンペア実行

手段132、および、入れ替え実行手段133を有する。ソフトウェア保守を実施する場合、ソフト配布手段112がソフトウェア資源管理データベース111のうち入れ替えるべきソフトウェアの新バージョン113と旧バージョン114を保守対象となる配布先計算機130のソフト受信エリア131にダウンロードする。旧バージョンのソフトウェアとは、配布先計算機にとって現行バージョンのソフトウェアを指す。新バージョンおよび旧バージョンのソフトウェアとして、それぞれがソースファイルまたはオブジェクトやライブラリおよびインストールスクリプトに加えてコンペアスクリプトを有している。そして、入れ替え実行手段133がソフト受信エリア131にダウンロードされた新バージョンのソフトウェアのインストールスクリプトを起動することによって旧（現行）バージョンから新バージョンへソフトウェアの入れ替えを実施する。もしも、この入れ替えが失敗した時には、入れ替え実行手段133がソフト受信エリア131にダウンロードされた旧バージョンのソフトウェアのインストールスクリプトを起動することによって旧（現行）バージョンのソフトウェアへの復旧を実施する。ここで、インストールスクリプトとは、それぞれのバージョンのソフトウェアから実行モジュールを生成するためのコンパイル・リンク・ロードコマンドを記述したものである。

【0019】本発明の特徴は、入れ替え実行手段133が動作する前に、コンペア実行手段132が旧バージョンのソフトウェアのコンペアスクリプトを起動することによって、旧バージョンのソフトウェアから疑似的に生成される実行モジュールと既に登録されてある実行モジュール（旧バージョンのソフトウェアの実行可能形式を指す）間の同一性を検証するものである。ここで、コンペアスクリプトとは、旧バージョンのソフトウェアから疑似的に生成される実行モジュールと既に登録されてある実行モジュール間を比較するためのコンペアコマンドを記述したものである。

【0020】（B）配布先計算機のDB構成例と処理の流れ

図2は、ソフト受信エリア131のデータベース構成例である。ソフト受信エリア131は、ソフト配布手段112によってダウンロードされたソフトウェアの新バージョン113と旧バージョン114が格納される記憶領域である。新バージョンのソフトウェアが「ソフトウェア名.N」のディレクトリ名称の構成下に、一方、旧バージョンのソフトウェアが「ソフトウェア名.B」のディレクトリ名称の構成下に格納される。例えば、入れ替えるソフトウェアとしてtaskAとtaskBが対象の場合、taskA.Nのディレクトリ下210にソフトウェアtaskAの新バージョン、taskA.Bのディレクトリ下220にソフトウェアtaskAの旧バージョン、taskB.Nのディレクトリ下230にソフトウェアtaskBの新バージョン、および、task

B.Bのディレクトリ下240にソフトウェアtaskBの旧バージョンが格納される。新バージョンと旧バージョンのソフトウェアは、それぞれがソースファイルまたはオブジェクトやライブラリおよびインストールスクリプトに加えてコンペアスクリプトを有している。例えば、ソフトウェアtaskAの新バージョンはオブジェクトtaskA.o211、インストールスクリプトap_load212、および、コンペアスクリプトap_comp213、ならびに、ソフトウェアtaskAの旧バージョンはオブジェクトtaskA.o221、インストールスクリプトap_load222、および、コンペアスクリプトap_comp223からなる。ここで、オブジェクトtaskA.o211とtaskB.o221間は新旧間で修正が加えられているので部分的に異なる。また、インストールスクリプトap_load212とap_load222間およびコンペアスクリプトap_comp213とap_comp223間はソフトウェア名称やコマンド名称等の変更が稀であるので同一のケースが多い。

【0021】図3は、コンペアスクリプトap_comp223のコンペアコマンドの記述例である。コマンド例としてUNIXシステムにおけるBourne shを用いている。コンペアスクリプトap_comp223が起動されると、301、302、303の順に解析され、結局、解析結果の「comp +P -o taskA taskA.o」が実行される。この「comp +P -o taskA taskA.o」は、オブジェクトtaskA.oから疑似的に生成した実行モジュールtaskAと既に登録されてある実行モジュールtaskA（現行バージョンのソフトウェアの実行可能形式taskAを指す）間の同一性を比較して、両者間の異なる部分を表示する。

【0022】図4は、コンペア実行手段132の処理の流れを示す。ステップ401は、ソフト受信エリアから現行バージョンのソフトウェア名の取り出しを行なう。例えば、ソフト受信エリア131のうちディレクトリ名がサフィックス「.B」を有するものを検索すれば、taskA.B220とtaskB.B240を取り出すことができる。ソフトウェア名が複数存在する場合には、プログラム名称の順に、以下のステップを実行していく。ここで以下では、ソフトウェア名としてtaskAを用いて説明する。ステップ402は、取り出したソフトウェア名を利用者に知らせるために画面等に表示する。例えば、taskA.Bからサフィックス「.B」を取り除いたものをソフトウェア名称として、画面に「ソフトウェア [taskA] のコンペア実行」と表示する。図5が画面に表示される内容の例である。ここでは、図5の501のように表示される。ステップ403は、取り出したソフトウェア名に対するコンペアスクリプトap_compを起動する。例えば、ソフトウェアtaskAのコンペアスクリプトap_comp223を起動する。コンペアスクリプトap_comp223を起動すると、結局、「comp +P -o taskA taskA.o」が実行される。この「comp +P -o taskA taskA.o」の実行結果は、オブジェクトtaskA.o221から疑似的に生成した実行

モジュールtaskAと既に登録されてある実行モジュールtaskA（現行バージョンのソフトウェアの実行可能形式taskAを指す）間が同一の場合には画面に何も表示されないが、逆に異なる場合には違いの詳細情報が画面に表示される。例えば、もしもこの実行結果が異なる場合には図5の502のように表示される。ステップ404は、取り出したソフトウェアのうちまだ未処理のものが存在するかを確認して、存在する場合にはステップ402に戻り、存在しない場合には終了する。例えば、取り出したソフトウェアのうちtaskBが未処理であるので、ソフトウェアtaskBに対してステップ402～403を処理する。取り出したソフトウェアに対して全て処理した後、本処理は終了する。

【0023】以上のようなコンペア実行手段132によって、入れ替えるべきソフトウェアを旧バージョンから疑似的に生成される実行モジュールと既に登録されてある実行モジュール（旧バージョンのソフトウェアの実行可能形式を指す）間の同一性を検証した後に、検証結果が良好の時には、新バージョンのソフトウェアに入れ替える処理に移行するが、検証結果が不良の時には、このような入れ替え処理を中止する。このように、同一性を検証しておくため、万が一、入れ替えるべきソフトウェアを旧バージョンから新バージョンに入れ替える処理が失敗した時でも、旧バージョンのソフトウェアへの復旧処理は、入れ替える前の実行モジュールに元通りに戻すことが可能となる。

【0024】【2】実施例2

実施例1の変形例として、コンペア実行手段132がコンペアスクリプトを実行する代わりに、コンペア実行手段132がインストールスクリプトを解析して、ソフトウェア種別に従ってコンペアコマンドを生成・実行する実施例を説明する。このようなコンペア実行手段は、インストールスクリプトを元にコンペアコマンドを生成・実行するので、コンペアスクリプトを作成するのがユーザにとって面倒な場合に有効となり、コンペアスクリプトを準備する必要がなくなる。なお、ハードウェア・ソフトウェア構成は図1と同様であり、ソフト受信エリアのデータベース131は、図2においてコンペアスクリプトap_comp213と223がない場合の構成となる。

【0025】図6は、インストールスクリプトap_load222のインストールコマンドの記述例である。コマンド例としてUNIXシステムにおけるBourne shを用いている。インストールスクリプトap_load222が起動されると、601、602、603、604の順に解析され、結局、解析結果の「reload +P taskA 100 +L taskA.o -l 10」が実行される。この「reload +P taskA 100 +L taskA.o -l 10」は、オブジェクトtaskA.oのリンクを実行することによって実行モジュールtaskAがタスク番号100の優先度10で主メモリ内に生成される。

【0026】図7は、コンペア実行手段132がインス

トールスクリプトを解析して、ソフトウェア種別に従ってコンペアコマンドを生成・実行する処理の流れを示す。図7におけるステップ401、402、および、404は、図4と同様の処理である。ステップ701は、取り出したソフトウェア名に対するインストールスクリプトap_loadを解析する。例えば、ソフトウェアtaskAのインストールスクリプトap_load222を解析する。ソフトウェア種別によって生成すべきコンペアコマンドが異なるので、ここでは、インストールスクリプトap_load222のうちでソフトウェア種別を識別するための変数KINDを設けている。インストールスクリプトap_load222を解析すると、変数KINDの値がtaskであるので「comp +P」を生成して、変数NAMEの値がtaskAであるので「-o taskA」を生成して、変数OBJECTの値がtaskA.oであるので「taskA.o」を生成する。これらを並べた結果として、コンペアコマンド「comp +P -o taskA taskA.o」を生成する。ステップ702は、生成したコンペアコマンドを起動する。例えば、コンペアコマンド「comp +P -o taskA taskA.o」を起動する。ステップ701と702は、結果的にはステップ403と同様の内容となる。

【0027】以上のようなコンペア実行手段によって、インストールスクリプトを元にコンペアコマンドを生成・実行するので、コンペアスクリプトを準備する必要がなくなり、ユーザの負担軽減およびディスク容量の削減につながる。その反面、コンペアコマンドを解析して生成するのに要する時間が増加する。

【0028】【3】実施例3

実施例1の変形例として、コンペア実行手段132がコンペアスクリプトを実行する代わりに、コンペア実行手段132がインストール先を変更した形式でインストールスクリプトを実行後に、変更したインストール先に生成された実行モジュールと現行バージョンの実行モジュール間でコンペアを実行する実施例を説明する。このようなコンペア実行手段は、ソフトウェア種別がプロセスやテキストデータの場合に有効であり、疑似的に生成した実行モジュールと現行バージョン間の実行モジュール間を比較するコマンドが提供されていない場合に必要となる。なお、ハードウェア・ソフトウェア構成は図1と同様であり、ソフト受信エリアのデータベース131は、図8に示す通りである。

【0029】図8のソフト受信エリア131は、入れ替えるソフトウェアとしてprocessAとprocessBが対象の場合を例にしているので、図2のデータベース構成のうちで、taskAがprocessAに、および、taskBがprocessBに置き換わった構成となる。しかも、コンペアスクリプトap_compを含んでいない。すなわち、processA.Nのディレクトリ下810にソフトウェアprocessAの新バージョン、processA.Bのディレクトリ下820にソフトウェアprocessAの旧バージョン、processB.Nのディレクトリ下

830にソフトウェアprocessBの新バージョン、および、processB.Bのディレクトリ下840にソフトウェアprocessBの旧バージョンが格納された構成である。ソフトウェアprocessAの新バージョンはオブジェクトprocessA.o811およびインストールスクリプトap_load812、ならびに、ソフトウェアprocessAの旧バージョンはオブジェクトprocessA.o821およびインストールスクリプトap_load822からなる。

【0030】図9は、インストールスクリプトap_load822のインストールコマンドの記述例である。インストールスクリプトap_load822が起動されると、901、902、903、904の順に解析され、結局、解析結果の「cc -o /usr/local/bin/processA processA.o」が実行される。この「cc -o /usr/local/bin/processA processA.o」は、オブジェクトprocessA.oのリンケージを実施することによって実行モジュールprocessAがインストール先の/usr/local/binに生成される。

【0031】図10は、コンペア実行手段132がインストール先を変更した形式でインストールスクリプトを実行後に、変更したインストール先に生成された実行モジュールと現行バージョンの実行モジュール間でコンペアを実行する処理の流れを示す。図10におけるステップ401、402、および、404は、図4と同様の処理である。ステップ1001は、インストール先に正式に実行モジュールを生成せずに仮の実行モジュールを必要とするために、取り出したソフトウェア名に対するインストールスクリプトap_loadをインストール先を変更して実行する。例えば、ソフトウェアprocessAのインストールスクリプトap_load822のうちで指定されたインストール先の変数INSTDIRの値を、（カレントディレクトリを意味する）に変更した結果、すなわち、「cc -o ./processA processA.o」を実行する。この結果、カレントディレクトリ（図8の820）に実行モジュールprocessAが生成される。ステップ1002は、仮に生成した実行モジュールと正式な現行バージョンの実行モジュール間で比較検証を行なう。例えば、UNIXシステムにおけるファイル比較コマンドcmpを利用して、「cmp ./processA /usr/local/bin/processA」を実行することによって比較検証を行なう。正式な現行バージョンの実行モジュールの所在は、インストールスクリプトap_load822の変数INSTDIRの値（/usr/local/bin）である。このように、ステップ1001と1002は、インストールスクリプトap_load822を元に、仮の場所に生成させた実行モジュールと正式な現行バージョンの実行モジュール間の比較検証を行なうことができる。

【0032】〔4〕各実施例の補足

（A）コンペア実行手段の起動タイミング

実施例1～3のそれぞれにおいて、コンペア実行手段132の起動タイミングとして、（1）配布先計算機に入れ替えるべきソフトウェアをダウンロード後、（2）配

布先計算機を立ち上げ後、（3）配布先計算機を立ち下げ時、（4）ユーザの指定した日付と時刻、（5）CPU負荷が低い時、等を設けておき、配布先計算機の特徴に応じて使い分ける。上記の（1）は、ソフト配布手段112が入れ替えるべきソフトウェアをダウンロードした後にコンペア実行手段132に実行指令を与えれば実現される。上記の（2）は、配布先計算機の立ち上げのイニシャル処理の中にコンペア実行手段132に実行指令を要求するコマンドを記述することによって実現される。上記の（3）は、配布先計算機の立ち下げのシャットダウン処理の中にコンペア実行手段132に実行指令を要求するコマンドを記述することによって実現される。上記の（4）は、ソフト配布手段112が入れ替えるべきソフトウェアをダウンロードした後にコンペア実行手段132に指定日付・時刻での実行指令を与えれば実現される。指定日付・時刻での実行指令にはUNIXシステムでのatコマンドがある。上記の（5）は、ソフト配布手段112が入れ替えるべきソフトウェアをダウンロードした後にコンペア実行手段132に低優先度指定での実行指令を与えれば実現される。低優先度指定での実行指令にはUNIXシステムでのniceコマンドがある。このような起動タイミングの使い分けによって、適切なタイミングでソフトウェア保守を実施できるので、円滑な計算機システムの運用を達成できる。

【0033】（B）コンペア実行から入れ替え実行への移行

実施例1～3のそれぞれにおいて、コンペア実行手段132の処理終了後から入れ替え実行手段133への移行の仕方として、（1）比較検証結果が全て一致の時に限り入れ替え実行へ移行、少なくとも1つの不一致がある時には入れ替え実行を中断する、（2）比較検証結果が一致のソフトウェアのみを入れ替え実行手段が実施し、不一致のソフトウェアを無視する、等の手段を設けて、配布先計算機の特徴に応じて使い分ける。このような移行方法の使い分けによって、計算機システムの信頼度に従ったソフトウェア保守を実施できる。

【0034】

【発明の効果】以上のような、コンペア実行手段によって、旧バージョンのソフトウェアから疑似的に生成される実行モジュールと既に登録されてある実行モジュール（旧バージョンのソフトウェアの実行可能形式を指す）間の同一性を検証しておくため、万が一、該ソフトウェアを旧バージョンから新バージョンに入れ替える処理が失敗した時でも、旧バージョンのソフトウェアへの復旧処理は、入れ替える前の実行モジュールに元通りに戻すことが可能となる。

【0035】このような元通りに戻せられるという高信頼なソフトウェア保守の実現によって、常にソフトウェアは正常な動作状態を維持することができ、ソフトウェアシステムの安全な稼働を保証させることができる。

【図面の簡単な説明】

【図1】ソフトウェア保守システム構成の一例である。

【図2】ソフト受信エリアのデータベース構成（タスクの対象）の一例である。

【図3】タスク向けコンペアスクリプトのコマンド記述の一例である。

【図4】コンペア実行手段のコンペアスクリプト起動処理の一例である。

【図5】画面への表示内容の一例である。

【図6】タスク向けインストールスクリプトのコマンド記述の一例である。

【図7】コンペア実行手段のコンペアコマンド生成・実行処理の一例である。

【図8】ソフト受信エリアのデータベース構成（プロセスの対象）の一例である。

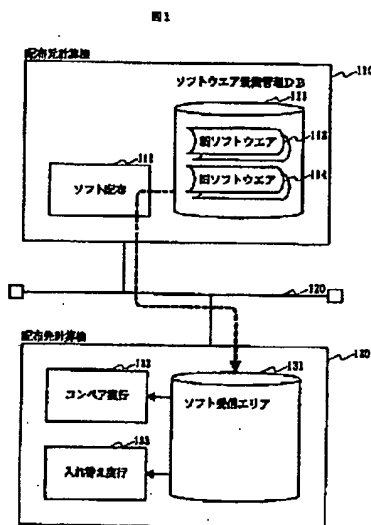
【図9】プロセス向けインストールスクリプトのコマンド記述の一例である。

【図10】コンペア実行手段の仮の実行モジュール生成とコマンド実行処理の一例である。

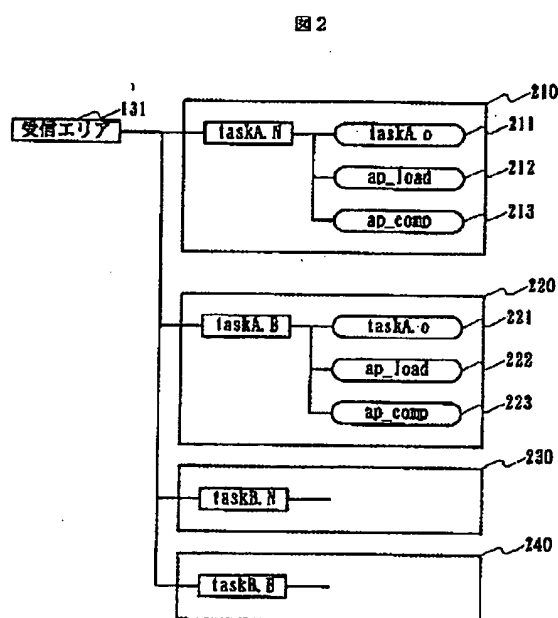
【符号の説明】

110…配布元計算機、120…ネットワーク、130…配布先計算機、111…ソフトウェア資源管理データベース、112…ソフト配布手段、113…新バージョンのソフトウェア郡、114…旧バージョンのソフトウェア郡、131…ソフト受信エリア、132…コンペア実行手段、133…入れ替え実行手段。

【図1】

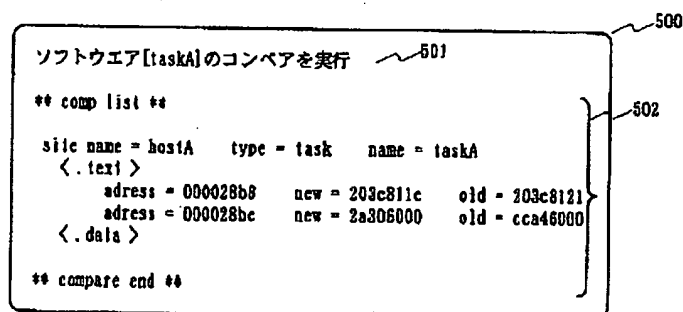


【図2】



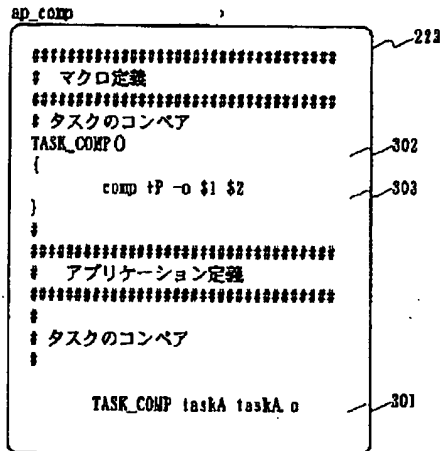
【図5】

図5



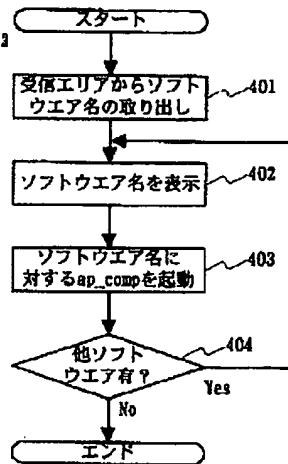
【図3】

図3



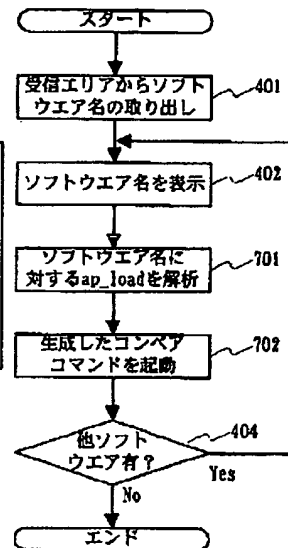
【図4】

図4



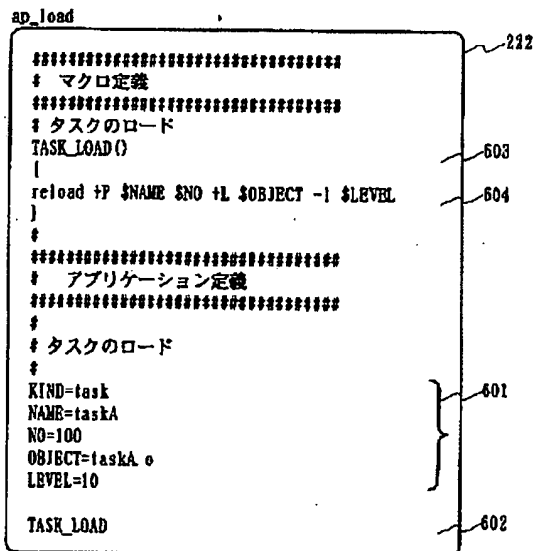
【図7】

図7



【図6】

図6



【図8】

図8

